7-1-1995

A Simpler robotics controller for a PhotoCD

Jeffrey Jackson

A Simpler Robotics Controller For A PhotoCD Jukebox

by

Jeffrey Jackson

A Graduate Paper Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Electrical Engineering

Approved By:

Professor Signature Illegible

Department of Electrical Engineering
College of Engineering
Rochester Institute of Technology
Rochester, New York
July, 1995

Table of Contents

ACKNOWLEDGMENTS 4

ABSTRACT 5

OBJECTIVE 6

INTRODUCTION 7

Drive Selection 8

Original Robotics Control Electronics 10

HARDWARE 11

The Controller 11

The Texas Instruments TPIC2801 11

Complications 12

SOFTWARE 14

PL/M 51 14

Host Command Interpreter 14
Table 1. Command Block Format 15
Table 2. Valid Commands 15
Table 3. Acknowledge Block Format 16
Table 4. Acknowledge Return Codes 16

Parser Routine 17

Robotics Control 17
Table 5. Command Byte Bit Functions 19

Serial Data Output 19

Functional Status Feedback 20

CONCLUSION 21

PSEUDOCODE LISTING 22

PLATES 25

Plate I. Engineering Model PhotoCD Jukebox 26

Plate II. Cassette With PhotoCD Loaded 27

Plate III. Modified Pickup Arm 28

Plate IV. Underside Of Pickup Arm 29

Plate V. Right Grip Mechanism, Enlarged 30

Plate VI. Original Audio Compact Disc Player 31

Plate VIII. Modified And Unmodified CD ROM Drives 32

Plate IX. Intelligent Driver Device 33

Plate X. Modified Pickup Driver Board 34

REFERENCES 35

Acknowledgments

The author wishes to thank Dr.Unnikrishnan and Dr. Reddy for their support, patience and guidance, James Schueckler (Eastman Kodak Company) for his advice and insights over the years, and John McGlone (Eastman Kodak Company) for his assistance in preparing the plates and the use of his facilities. The author also expresses his gratitude and appreciation to his long-suffering wife and family for their support in this effort.

Abstract

In the current industrial and economic environment, research and development funds are very limited. There is strong motivation from management to build upon existing hardware and software whenever possible. The premise is that time-to-market will be reduced and more utility can be gleaned from previously spent development moneys. This paper is the combination of a case study of the re-engineering of an existing product and the design and implementation of the next step in the process, whereby parts count is reduced to cut costs and increase reliability.

Objective

Goals sought are reduced parts count , reduced cabling and interconnections (and the increased reliability/maintainability that is attendant with such reductions), and development of robotics control software that is easily modified and supported. If possible, it should not be apparent to the host computer whether the jukebox is the new or old model. A more recent vintage microcontroller, the Intel 8031 with 8 Kbytes of external RAM and 8 Kbytes of external EPROM, was selected to control the jukebox. While not a new device, it is common and inexpensive. New 8031-family microcontroller devices are being introduced to the marketplace on an ongoing basis by several manufacturers, incorporating an increasing number of features in a single package and delaying obsolescence of the underlying technology.

Introduction

This work is based on a larger project, the PhotoCD jukebox. The PhotoCD jukebox is a 100-disc storage unit with a single CD ROM drive. Any individual disc can be automatically retrieved by the robotics and placed in the drive for playback. The discs are stored in individual sliding "cassettes", minimizing media wear, since the robotics mechanism never directly contacts the disc, only the cassette. The cassettes, in turn, are stacked in two removable 50-cassette magazines. Thus the user may have available on-line over 60 Gbytes of storage, or its equivalent, over 10,000 PhotoCD images, in a unit no larger than a seventeen-inch video display monitor.

The PhotoCD jukebox has two host interfaces. A Small Computer System Interface (SCSI) handles data transfers between the CD ROM drive and the SCSI host adapter card in the host computer. A second interface is used to control the robotics. This is a simple serial interface using RS232 signal levels and a proprietary command set. The robotics operate almost completely independently from the CD ROM drive, creating challenges in maintaining "synchronization" between the CD ROM drive and the robotics, and in monitoring status of the two separate subsystems from the host computer. A single interface to the jukebox via SCSI is certainly preferable, but was beyond the scope of this project.

A large part of the problem with converting to SCSI control of the robotics lies with the present state of the SCSI specification itself. As the SCSI standard evolves, jukebox control commands and status sense messages will undoubtedly be incorporated and become fixed components of the specification. Until this happens, manufacturers are content to maintain a separate serial interface for robotics control, thus avoiding the issue entirely.

There is much risk in developing any SCSI device which requires proprietary extensions to the SCSI specification. If the SCSI standards committee settles on a command set different from the proprietary extensions, the device manufacturer is saddled with a product which is instantly obsolete.

Drive Selection

The PhotoCD jukebox was built from a commercially available Compact Disc Audio (CDA) jukebox. The original CD drive in the jukebox was not compatible with PhotoCD, since PhotoCD requires compliance with Compact Disc Read Only Memory eXtended Architecture (CD ROM XA), mode 2, form 1, multi-session specification. This is commonly referred to as "Orange Book" compliance, because the handbook defining the specification has an orange colored cover.

To make the jukebox PhotoCD compatible, the existing drive had to be replaced with a suitable PhotoCD-capable unit. At the time, in early 1992, there were very few CD ROM drives on the market which were PhotoCD compatible. To further complicate matters, those drives that could properly read a PhotoCD at all would only play back the first session (all were first-session-only drives). This meant that all images recorded on the disc would have to be written in one continuous operation. If the creation of the disc were to be broken up into multiple operations, only those written in the first session would be readable (hence the term "multi-session" for those drives capable of looking beyond the lead out track for the table of contents of another recording session, and "single-session" for drives that can only "see" the first table of contents).

In terms of film rolls, the user would have to completely fill the disc with four 36 exposure rolls of negatives (at one sitting) to most efficiently utilize disc space. While not convenient, it was a suitable work-around until multi-session drives became available. Since there was ongoing dialogue with the various drive manufacturers, it was possible to know which vendors were interested in the multi-session drive market. The drive selected had to be from a vendor planning to transition to multi-session in the future, without making wholesale changes in the physical form factor of the drive, something which would cause additional costs in re-design time and money.

Host compatibility was another factor to be considered in selecting the drive. At the time, interest was mainly in Apple Macintosh, Sun Microsystems, and IBM PC (and

compatible) platforms. It was desirable to find a CD ROM drive for which device driver software for all three platforms was already available, since time and development funds were very limited and there was little commercial motivation to develop device drivers for other manufacturer's products.

The physical interface also had to be common across all three platforms. This requirement indicated a drive with a Small Computer System Interface (SCSI). As stated previously, robotics control is via ASCII commands transmitted from the host via an RS232 serial link. In the modified jukebox, this was retained. Robotics control via serial interface is nearly universal in the mass storage jukebox market. To avoid breaking "new" ground with this product, intended for the low-end of the business/professional market, the serial interface was retained.

There was only one CD ROM drive which met all of the above criteria, the Sony CDU541. This was a single-session drive with a multi-session stable mate (the CDU561) soon to be introduced. Unfortunately, the mechanical portion, or transport, of this drive was larger and physically configured differently than the CDA drive it replaced. The drive, the jukebox, or as it turned out, both, had to be altered to squeeze the larger transport into the space left by removing the original audio-only unit.

To modify the CDU541, it was disassembled, cable assemblies were relocated, and the transport base plate was cut to remove approximately 3/4 inch from the rear. The outer case was discarded and the interface board mounted below the transport, rather than above, as it was in its original configuration. This was necessary to allow access to the transport from above, to load the discs from the magazines.

The pickup arm, which slides the cassettes out of the magazines and places them on the CD transport, was designed (literally) around the original CDA transport. It proved impossible to modify the original pickup arm to work properly with the new transport, so it was discarded and a new pickup arm, which supported the cassette from above was designed.

The jukebox thus modified was sufficiently robust to loan to software authors so they could begin application software development. Thirteen identical units were constructed. A fourteenth was scrapped as it was used as a prototype in early

development and could not be rebuilt to match the other units closely enough to be supported in the field. It is this fourteenth unit that I have used for the development work for my paper. It will never be capable of actually loading and reading a disc, since the pickup arm is damaged beyond repair and I do not have a modified CDU541 to install in it. The pickup assembly (which carries the pickup arm), the robotics mechanism, and the magazines and cassettes are all intact, however.

Original Robotics Control Electronics

The electronics which control the robotics in the original jukebox apparently evolved over many generations of engineering changes and functional enhancements. They are built around a very mature microcomputer family, the Rockwell 6502. This is the same family used in the original Apple personal computers. While perfectly adequate for the task, it requires a large number of support devices and "glue" logic. On the pickup controller card itself, there are 4 LSI devices, 14 CMOS and LSTTL ICs, and well over 100 discrete parts. The driver board for the pickup stepper motor and the cassette gripper motors has more CMOS ICs, driver transistors, discrete suppression networks (to clamp transients from the inductive loads), and a large number of discrete parts for input pull-up and level matching . The pickup driver board is connected to the pickup controller board and the motors it drives by 5 flat-cable assemblies.

Hardware

The Controller

For demonstration purposes, the project is built using the Control-R II from Cottage Resources Corporation. This is a compact single board controller built around the Intel 8031 single chip microcontroller. It has 8192 bytes of RAM (read/write), 8192 bytes of EPROM, and a Maxim, Inc. single-chip RS232 interface.

The Texas Instruments TPIC2801

Almost the entire pickup driver board has been replaced with one specialized integrated circuit driver, the Octal Intelligent-Power Switch With Serial Input, the TPIC2801, from Texas Instruments. This device simultaneously sinks one ampere from each of its eight outputs (eight amperes total current sinking capability), and has internal suppression circuitry which clamps inductive-load-induced voltage transients to 35 volts, peak. It is a serial device, requiring only five control lines from the CPU-- Serial Input, Serial Output, Serial Input/Output Enable, Serial Clock, and Reset. A serial output permits very easy daisy-chaining of multiple devices. Although not used in this project, this feature will simplify future expansion or product enhancements.

The condition of each of the individual output stages may be monitored by the host via a fault-conditions check function. Open circuit conditions on any output may be checked by programming the output OFF, then clocking the device to read back the status byte from the serial output pin. A logical low level at that output stage's corresponding bit indicates that the load has become disconnected from the output and is therefore not pulling it up. Current overload conditions are revealed in a similar fashion. An output is turned ON in this case. The device is clocked to return the status byte to the CPU via the serial output pin. A current overload condition is indicated by a logical high level at the corresponding status bit, since the device's current overload protection circuitry has turned off the drive for this pin, due to the fault condition. Function of the system powered by this device may thus be easily and quickly verified by saving a copy of the control byte before it is sent to

the device, clocking back the status byte from the device, performing an exclusive OR, and checking for any set bits in the result. A one in any bit position indicates a fault at the corresponding output drive pin.

An added bonus of using such a compact device is that it permits wiring to the inductive loads to be very short, since the driver may be mounted close to the load. This is important in terms of reducing electromagnetic interference (EMI). The inductive transients are clamped locally, without being conducted over a lengthy wiring harness where they would be radiated from a longer antenna with a much larger loop area. A future enhancement might be to use two TPIC devices - one to drive the pickup arm lift stepper motor (and mounted in close proximity to it), and a second TPIC mounted directly on the pickup arm, to drive the cassette gripper motors. The EMI from the permanent magnet cassette gripper motors is much less than that from stepper motors, which are notoriously noisy. Reliability might be enhanced, however, if the cassette gripper motors, which have brushes, were to be replaced with stepper motors, which are virtually maintenance free. As stated before, there would be almost no cost in harnessing changes, since the TPIC devices could be daisy-chained and use the same control lines from the CPU.

Complications

While working through the original schematic diagrams and during the hardware debugging process, several problems were discovered. The cassette gripper motors on the transport arm are used in a bi-directional fashion. This means that a unipolar drive device like the TPIC cannot be used. The original circuit used an L298 bipolar output H-switch to run the cassette gripper motors. This device is perfectly suitable but requires external transient suppression to protect the outputs from the voltage spikes fed back from the inductive loads. In addition, each output takes a separate output port pin from the microcontroller, and they are all used, since the Control R-II is configured with external memory, which takes up all of ports 0 and 2, and parts of port 3. To solve the problem of limited pins, I used the outputs from the TPIC to operate the L298. The disadvantage of using this approach is that the pickup driver board could not be totally eliminated, since I am using the L298 and all of the associated voltage transient suppression circuitry that is already present.

A second problem arose with the status feedback circuits. The jukebox had a series of 4000-series shift registers to output the command bits to the drivers and to read back the status bits from the various optical interrupters. The optical interrupters were wired in a common collector configuration, feeding a 15 volt signal from their emitters back to the CMOS shift register inputs. The shift registers (as well as the rest of the logic) operate on 5 volts, so the original design incorporated a discrete voltage divider network for each of the inputs, to bring the input signal into the safe operating range for the integrated circuit. The impedance of the divider circuit is 27 Kohms, a fairly high level to be connected to a harness that is nearly a foot long and is in the vicinity of power-carrying lines and inductive loads. The danger is that false signals can be received via crosstalk, when the input impedance is as high as it is with this present design.

My solution was to eliminate the CMOS devices entirely and read the status directly with the input port pins of the 8031. The optical interrupters were re-wired into a common emitter or "pull-down" configuration. The port 1 pins of the 8031 all have internal active pull-ups, eliminating the need for external discrete parts. When the optical interrupters are active (not blocked) they feed a signal back to the 8031 that is within Vce(sat) of ground potential. When blocked, no current is conducted, and the port pins are pulled up to a logical one level by internal Field Effect Transistor current sources. While I have not benchmarked the new design against the original one, I have had poor luck in the past routing un-buffered machine control signals through lengthy harnesses into high impedance CMOS integrated circuit inputs. Even when the inputs are Schmitt triggers (these were not, in the original circuit) and there is some hysteresis to help prevent reception of false signals, it is possible to have intermittent errors that are very difficult to trace, since applying a test probe often provides enough termination (lowers the impedance) that the transient which is causing the problem disappears. This leaves the designer with a circuit that only works with oscilloscope probes attached, which is of limited usefulness. It should be noted, however, that the original circuit in question is currently used in a commercially successful product, with no problems that are known to date, so apparently the designer got everything right, in this particular case.

Software

PL/M 51

The controller software is written in Intel's PL/M 51, a block-structured, high-level language. It is divided into small function-specific blocks, each of which performs a limited number of tasks. This modularity permits easy debugging and software maintenance, since each function can be developed independently, using a simple program stub to feed it the proper parameters and display returned values. Descriptive procedure and variable names are used throughout the program, in an effort to make it read in a "plain English" manner.

The main program loop is less than a page. It waits in a loop for valid commands from the host, then calls the appropriate procedures. It returns status to the host and resumes waiting.

Host Command Interpreter

Commands are sent from the host in a variable-length Command Block. Command blocks from the host are received, converted from ASCII to hexadecimal, and stored in an array. Once in the array, they are parsed and their various components are evaluated and acted upon. The command block format is mapped in Table 1, valid commands are tabulated in Table 2.

Offset	Length	Contents	Comments
0	1	STX	Start of TeXt (STX), hexadecimal 02h
1	2	00...99	Number of bytes, excluding ETX
3	2	00...99	Command code
5	n	Data	Command Arguments
6+n	1	ETX	End of TeXt, hexadecimal 03h

Table 1. Command Block Format

Code	Argument	Comments
00...99		Reserved
22	*rdd*	*r*: A = Absolute disc location reference, +/- = relative location reference *dd*: Number of disc to load
24		Return disc

Table 2. Valid Commands

When the host software sends a command block, it waits for confirmation that the command was executed successfully. Status is returned to the host in an acknowledge block. Acknowledge block format and valid returned codes are tabulated in tables 3 and 4, respectively.

Offset	Length	Contents	Comments
0	1	STX	Start of TeXt (STX), hexadecimal 02h
1	2	00...99	Number of bytes, excluding ETX
3	2	00...99	Command code being acknowledged
5	2	00...99	Return code, 00 = command successful
7	n	Data	Acknowledge arguments
7+n	1	ETX	End of TeXt, hexadecimal 03h

Table 3. Acknowledge Block Format

Code	Meaning
00	Command executed successfully
01	Unknown Command
02	Improper command format
03	Jukebox busy
30	Error pulling cassette from magazine
31	Error returning cassette to magazine
32	Pickup arm lift failure

Table 4. Acknowledge Return Codes

Parser Routine

The parser proved to be fairly challenging. The original manufacturer's software developers had written a very robust parsing routine. It checked for valid command numbers, improperly constructed command blocks, and checked the total number of characters sent. Without the source code for these routines to reverse-engineer (which would probably violate copyright law and is therefore inappropriate) it was necessary to treat the original parser as a "black box". I did not know how it worked internally, but could observe the result of test data sent to it and attempt to duplicate responses to the same test data in my code. I was not entirely successful in my attempt, but if a teaming arrangement evolves with the original manufacturer, this will be resolved when code is exchanged.

Robotics Control

The robotics control portion of the program proved nearly as difficult as the parser, due to the limited number of I/O pins available when external memory is used with the 8031. The control program runs the picker arm stepper and the right and left cassette gripper motors. Stepper motors must have the proper phase windings energized at the correct time to either take a step clockwise or counter-clockwise. After a step is taken, the phase windings are left energized to "lock" the motor shaft in position.

Stepper motor patterns are stored in a table in EPROM with the control program. Successive entries in the table define values of legal "next steps". If an attempt is made to jump to an illegal step, the motor will not rotate properly. A pointer into this table is either incremented or decremented to rotate the motor in either direction. This pointer is actually an offset from the base of the step table, and is masked to three bits so the table wraps every eighth step. i.e., when the pointer is incremented from 7 to 8, and the result is logically ANDed with a mask value of 07H, the result is 0, or the base address of the step table, since the carry to the "8 bit" is masked off. Similarly, underflows from the 0 count to 255 (negative 1) when the pointer is decremented are also masked off and the pointer is placed at the eighth step (position 7) of the table.

The Cassette Gripper Motors are similarly controlled by setting bits to enable the "H"- switch outputs in the proper pattern. Direction is reversed by 1's complementing the gripper motor's control bits, thereby reversing the direction of current flow through the motor.

A command byte is assembled from the stepper table information and gripper motor control bits as shown below. This is done through a process of logically ANDing and ORing the Command Byte with the control bits that are to be changed.

For example, to turn on the right gripper motor, without changing anything else, requires the following steps:

1) Fetch the most recent command byte

2) Clear the right gripper motor control bits by logically ANDing.

3) Set the proper right gripper motor control bits by logically ORing with, for example 20h, thereby setting only bit 5 of the command byte, leaving everything else intact.

Bit	Function
0	Stepper Phase 0
1	Stepper Phase 1
2	Stepper Phase 2
3	Stepper Phase 3
4	Right Grip Motor, Positive
5	Right Grip Motor, Negative
6	Left Grip Motor, Positive
7	Left Grip Motor, Negative

Table 5. Command Byte Bit Functions

Serial Data Output

Once the command byte has been assembled, it must be output serially to the TPIC device. Since the 8031's on-board serial port is already used to communicate with the host computer, a single-bit output port technique was used which did not need the internal serial port hardware. Fortunately, the predecessor of the 8031, the Intel 8048, did not have a true serial port at all and Intel applications engineers had developed clever ways to work around this shortcoming using software to shift and toggle the data. I was able to adapt their solution. It works as follows:

1) A copy of the most recent port output data is fetched from where it had been stored in memory at time of the last output and the serial data bit is cleared.

2) The command byte is logically ANDed with a mask to clear everything except the least significant bit. The command byte is then right-shifted one bit and stored.

3) The result is logically ORed with the output data byte, preserving previous output data.

4) The output data byte is now written to the port and the clock line to the TPIC device is toggle by XORing the serial clock bit twice with the appropriate mask.

Functional Status Feedback

The last item is status feedback from the jukebox mechanism. The jukebox is equipped with opto-interrupters and interrupter flags to monitor the position of various mechanical components. An opto-interrupter consists of a wavelength-matched infrared emitting diode (IRED) and phototransistor, mounted in a U-shaped plastic frame. When there is nothing blocking the IRED's radiation, the phototransistor is in a conducting state (saturated). An opaque plastic or metal flag is mounted on the mechanism to be monitored. When this flag enters the slot in the U-shaped frame, the beam is blocked and the phototransistor no longer conducts. As mentioned previously, the opto-interrupters have all been re-wired to a more popular common emitter configuration. Their collectors are wired directly to the input port pins of the microcontroller. A pair of these devices is mounted on the picker arm to check the position of the right and left cassette grippers. A third device on the picker arm is used to confirm that the picker arm is in the "home" or fully down position.

The stepper motor used to raise and lower the picker arm is monitored by a fourth opto-interrupter. The state of all of the opto-interrupters is checked in an identical fashion. The port is read and the resulting byte is logically ANDed with a status check bit in the appropriate position. A non-zero result indicates that the opto-interrupter is blocked.

Conclusion

This project was embarked upon as an exercise in reducing the cost and increasing the reliability of a product that was originally designed for a low-volume professional market. A product for such a market can generally command a higher price than one aimed at the consumer. The higher selling price affords the designer more latitude and flexibility in working toward a solution, since the pressure for reducing unit manufacturing costs is not quite so intense. If a jukebox for data storage is to penetrate the smaller business and higher end consumer market (with much higher production volumes), however, it must be priced more aggressively. Attaining such a price point cannot be achieved using the conservative techniques I have outlined in this paper. Compact Discs, with proper handling, will last practically forever since they are written and read using non-contact optical means. Eastman Kodak, for example, quotes a one hundred year longevity for their Writable CD media. The disc's surfaces, whether laser written or pressed, are fairly delicate, however, and must not be touched during retrieval, playback, or replacement in the stack. There is no readily apparent way to handle them carefully without resorting to a complex (expensive) mechanical solution. With so much of the manufacturing expense of a product such as this tied up in the mechanical portion, the electrical design engineer will be challenged by management to cut the share of the cost borne by the electronics to an absolute minimum. Toward this end, the entire control electronics suite could conceivably be built into a single Application Specific Integrated Circuit. The problem of having power devices and logic share the same die is presently being addressed by integrated circuit manufacturers and, in fact, microcontrollers with high power buffers co-located on the same die are currently available on a limited scale. Service costs would be greatly diminished with a truly single chip solution. Troubleshooting becomes a matter of checking the power supply and opto-interrupters and, if they are operating properly, replacing the single device. As always, the question is one of having sufficient projected sales volumes to justify the NRE necessary to execute a total re-design from the ground up.

Pseudocode Listing

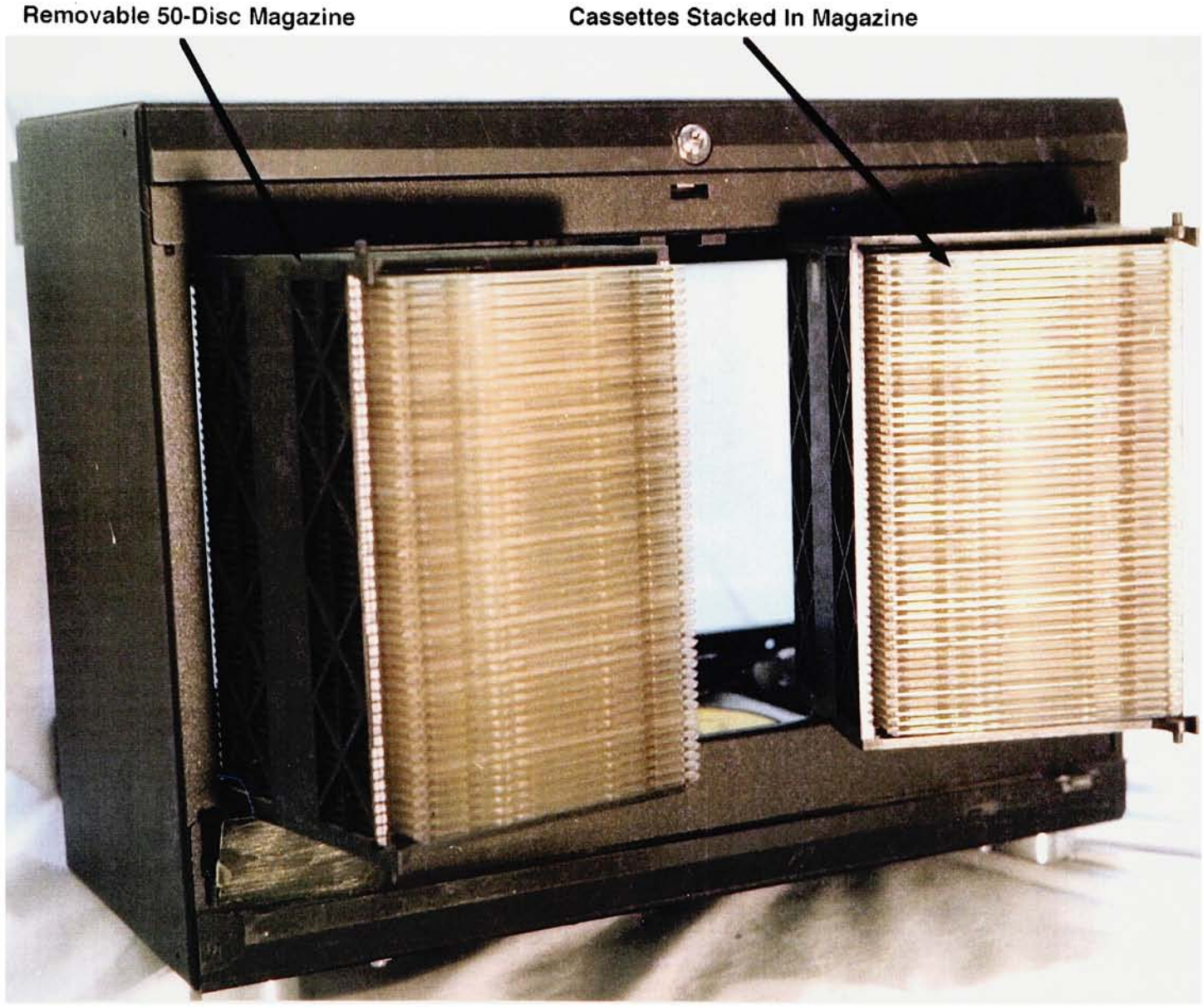
For more detailed information concerning the proprietary software incorporated in this project, please contact the author.

```
start:
initialize_variables; /* except current_slot and disc_loaded, stored in NVRAM;
if disc_loaded;
 then return_disc_to(current_slot);
go_home;
main_loop:
 fill(command_str_ptr,maximum_command_str_length, null);
 command_str_length = 0;
 get_char(command_str_ptr,command_str_length);

inner_loop:
get_command_str (command_str_ptr,command_str_length);
/* check for correct command format, then parse string for  the actual command
if valid_command_structure(command_str_ptr,command_str_length);
  then
    get_command_number(command_str_ptr,comand_str_length);
    execute_command(command_number);
  else send_error_message("Improper command format");
 end inner_loop;
end main_loop;


get_command_str:
  while not STX_detected;  /*wait for start of transmission
    get_char(command_str_ptr,command_str_length);
    end while;

  while not ETX_detected;   /* fill buffer until end of transmission is detected
  command_str_length = command_str_length + 1;
  command_str_ptr= command_str_ptr + 1;
  get_char(command_str_ptr,command_str_length);
  end while;
end get_command_str;
```

```
valid_command_structure (command_str_ptr, command_str_length):
buf byte based command_str_ptr;
 byte_count = dblasc_to_dec(buf[command_str_ptr]) /* read length of string
 if byte_count <> command_str_length then return false; /* compare to actual length
 else return true;
end valid_command_structure;

execute_command (command_str_ptr, command_str_length):
buf word based command_str_ptr;
 if buf[command_str_ptr + 3] = '22'
  then if disc_loaded
     then return_disc_to(current_slot);
     else
      current_slot = dblasc_to_dec(buf[command_str_ptr + 6]);
      load_disc_from(current_slot);
      disc_loaded = true;
 else if buf[command_str_ptr + 3] = '24'
  then return_disc_to(current_slot);
end execute_command;

load_disc_from (slot_number):
  position_picker_at(slot_number);
  if slot_number < 51;
   then pull_disc_from_left;
   else pull disc_disc_from_right;
  go_home;
end load_disc_from;

position_picker_at (slot_number):
  if slot_number > 50;
  then count = slot_number - 50;
  else count = slot_number;
  for step_index = 1 to count;
     step_index = step_index AND 07h; /* mask to least significant three bits
     step_pattern = step_table[step_index]; /* select proper stepper phase drive
pattern
     serial_output(step_pattern);
     if not picker_moved; /* XOR status with last reading to detect change
       then send_error('Picker movement error');
     end:
 end position_picker_at;
```

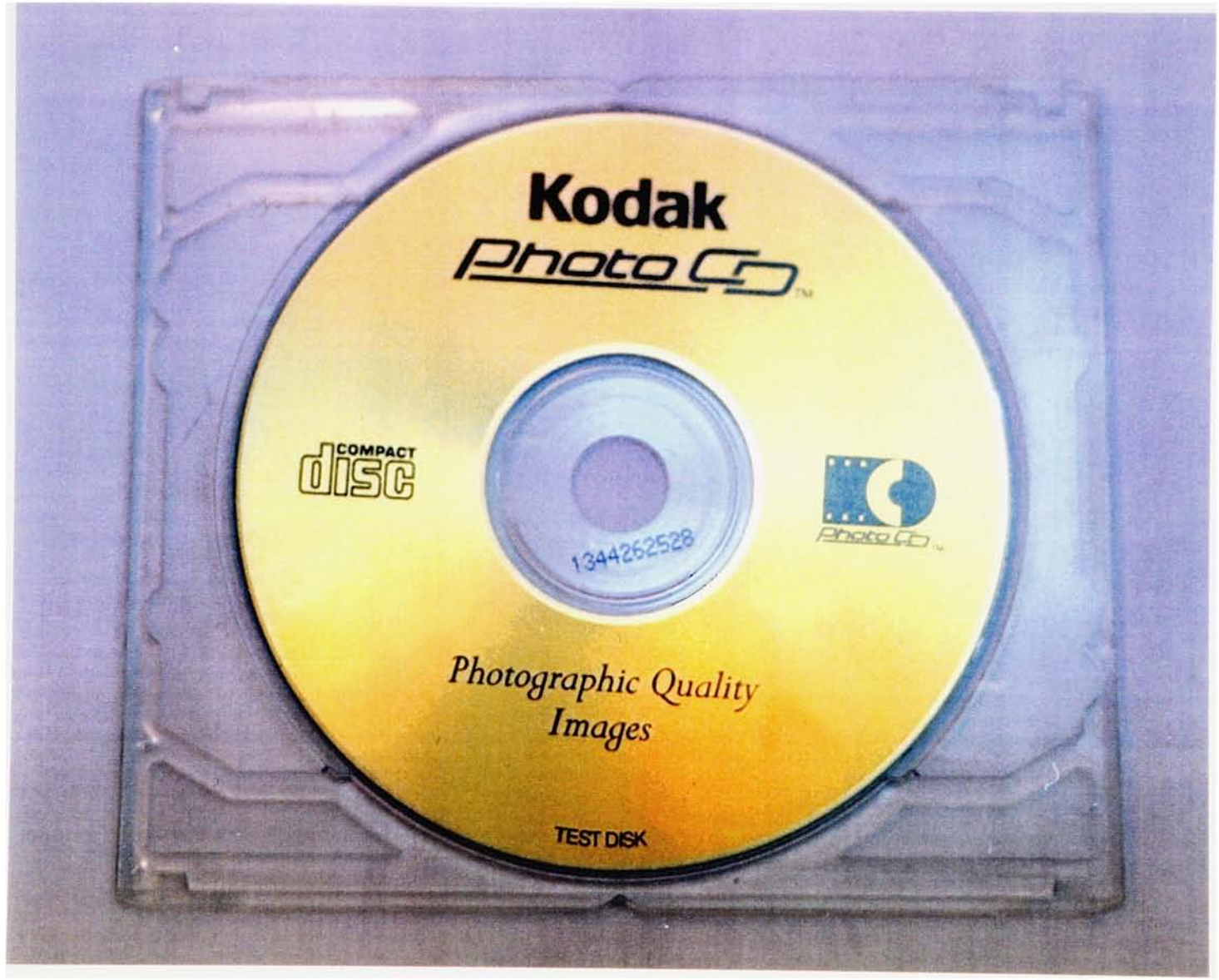
```
go_home:      /* send picker to home position
step_index = 0;
while not home_flag_detected ;
   step_index = step_index AND 07h; /* mask to least significant three bits
   step_pattern = step_table[step_index]; /* select proper stepper phase drive
pattern
   serial_output(step_pattern);
   if not picker_moved; /* XOR status with last reading to detect change
     then send_error('Picker movement error');
   step_index = step_index - 1;
  end;
end go_home;

return_disc_to(slot_number);
 position_picker_at(slot_number);
 if slot_number < 51;
  then push_disc_to_left;  /* slide cassette back into left magazine
  else push_disc_disc_to_right; /* slide cassette back into right magazine
 go_home;
end return_disc_to;
```

Plate I. **Engineering Model PhotoCD Jukebox**



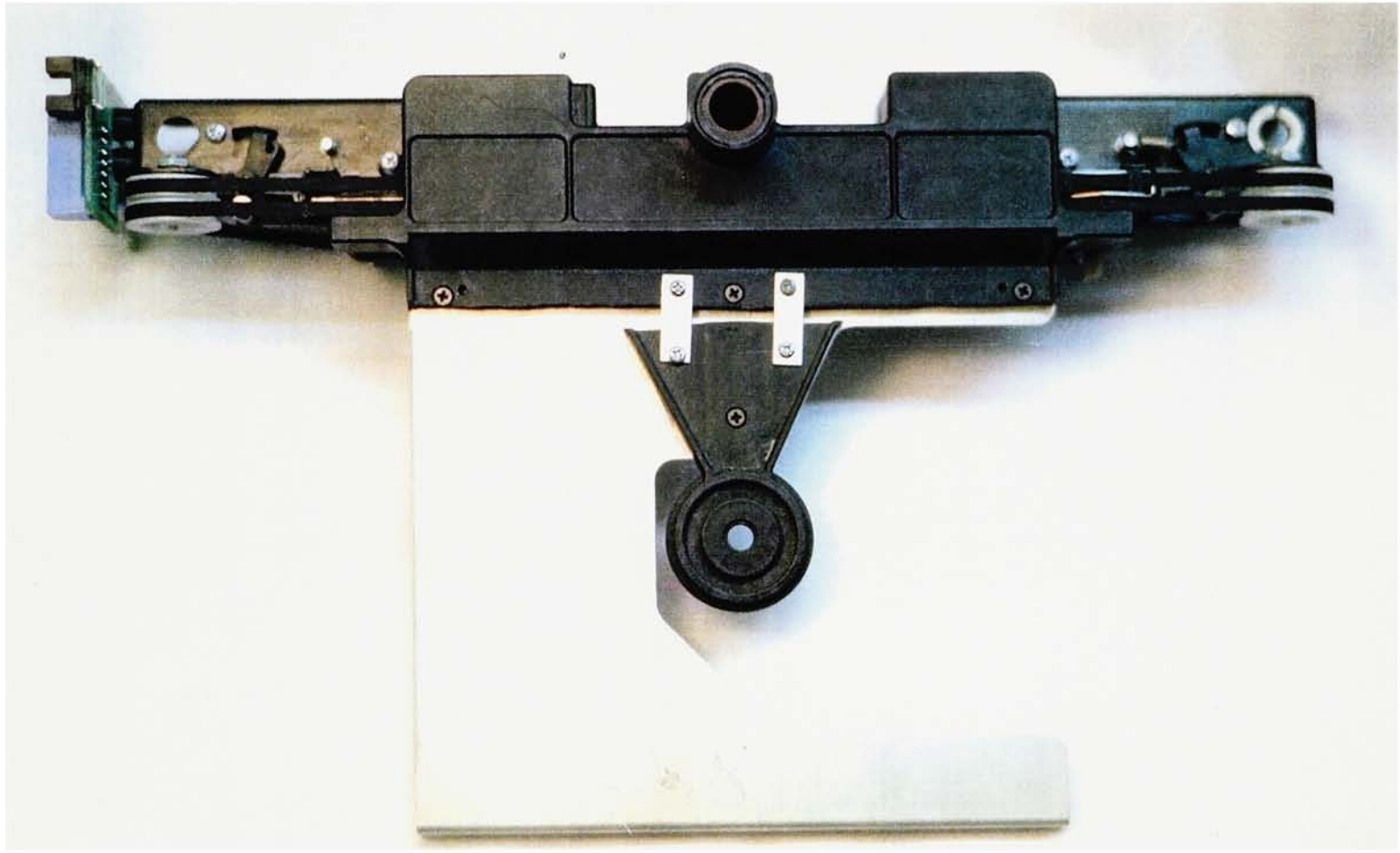
Engineering Model PhotoCD Jukebox

Plate II. Cassette With PhotoCD Loaded



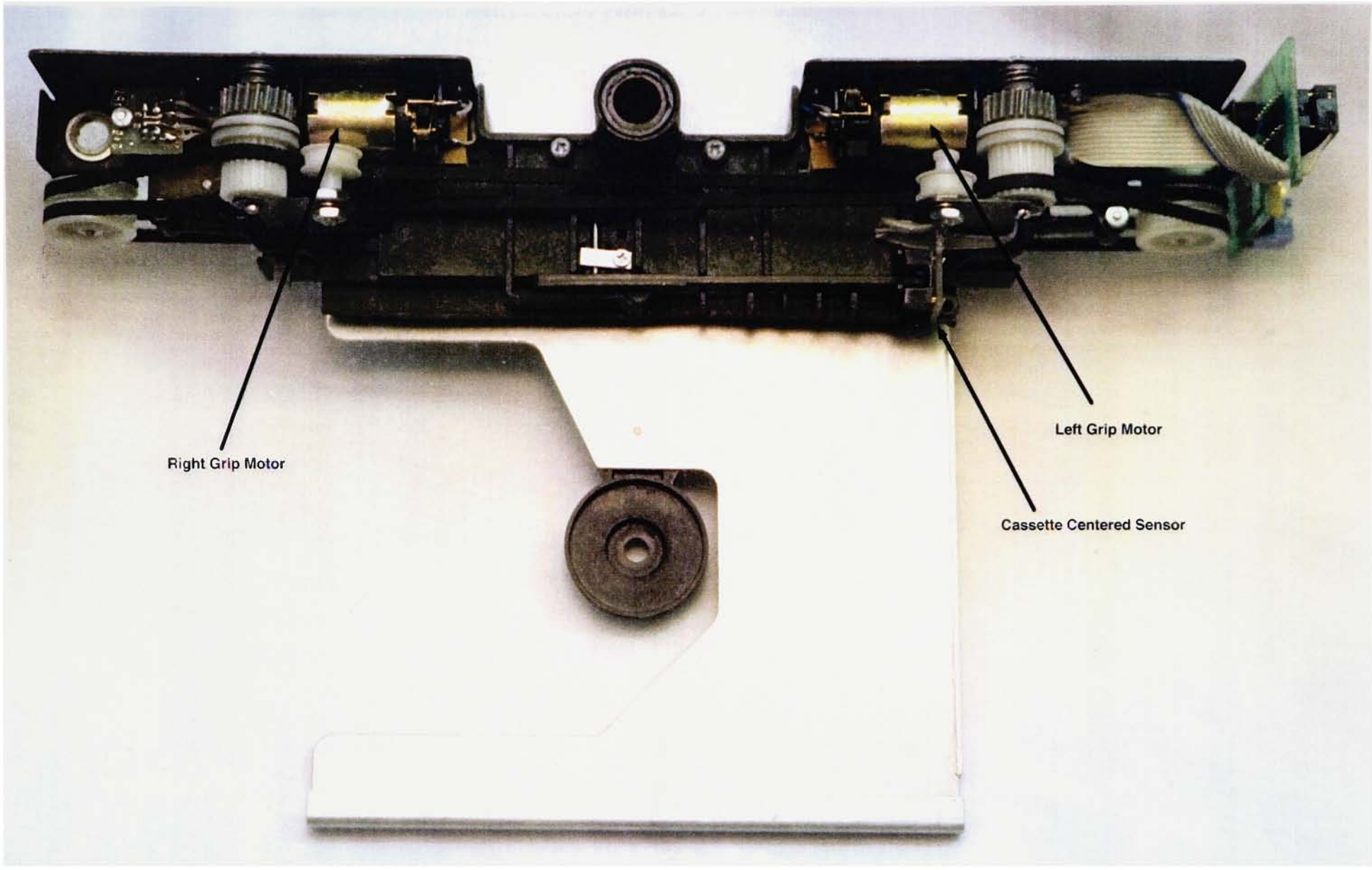
Cassette With PhotoCD Loaded

Plate III. Modified Pickup Arm



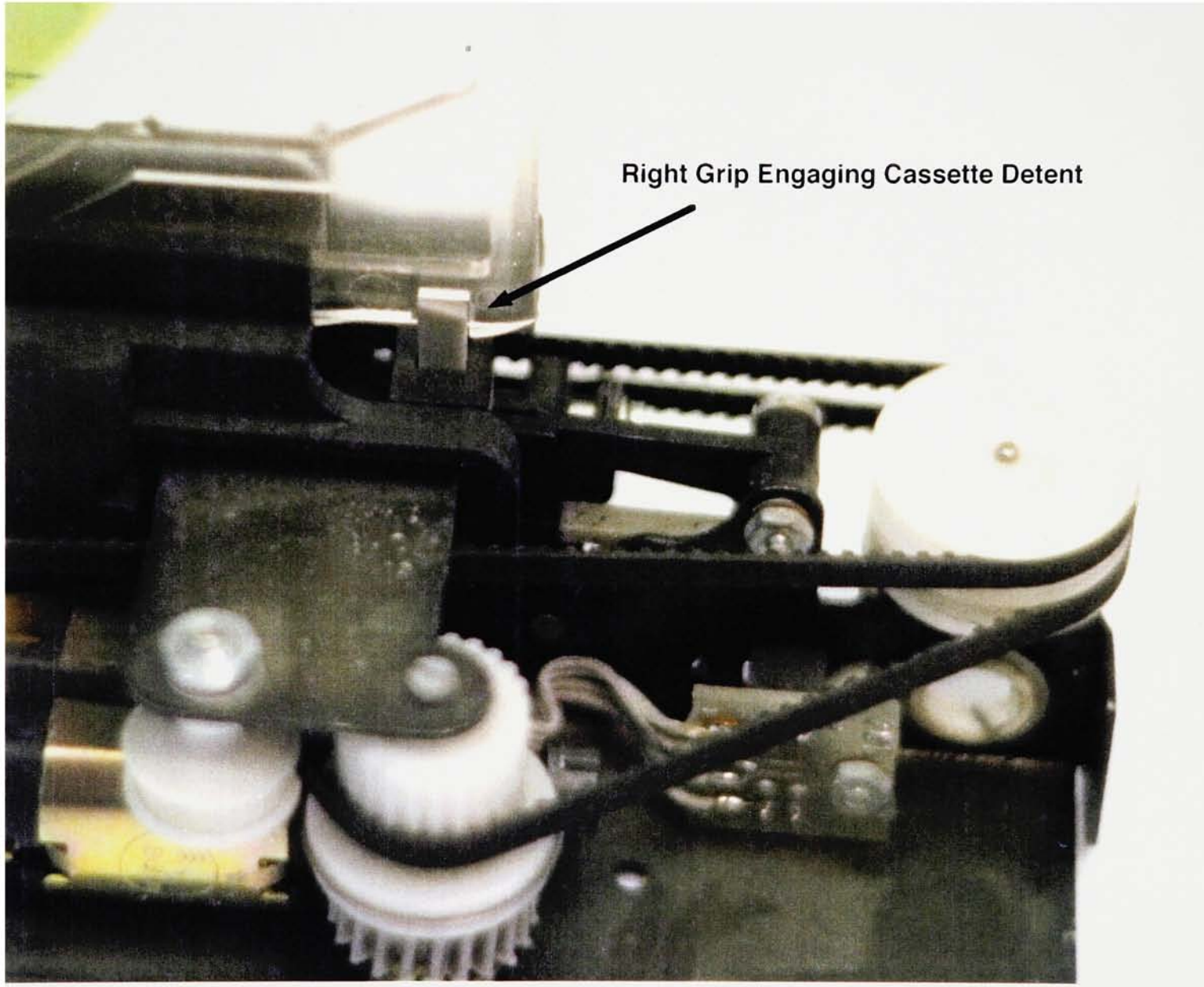
Modified Pickup Arm

Plate IV. Underside Of Pickup Arm



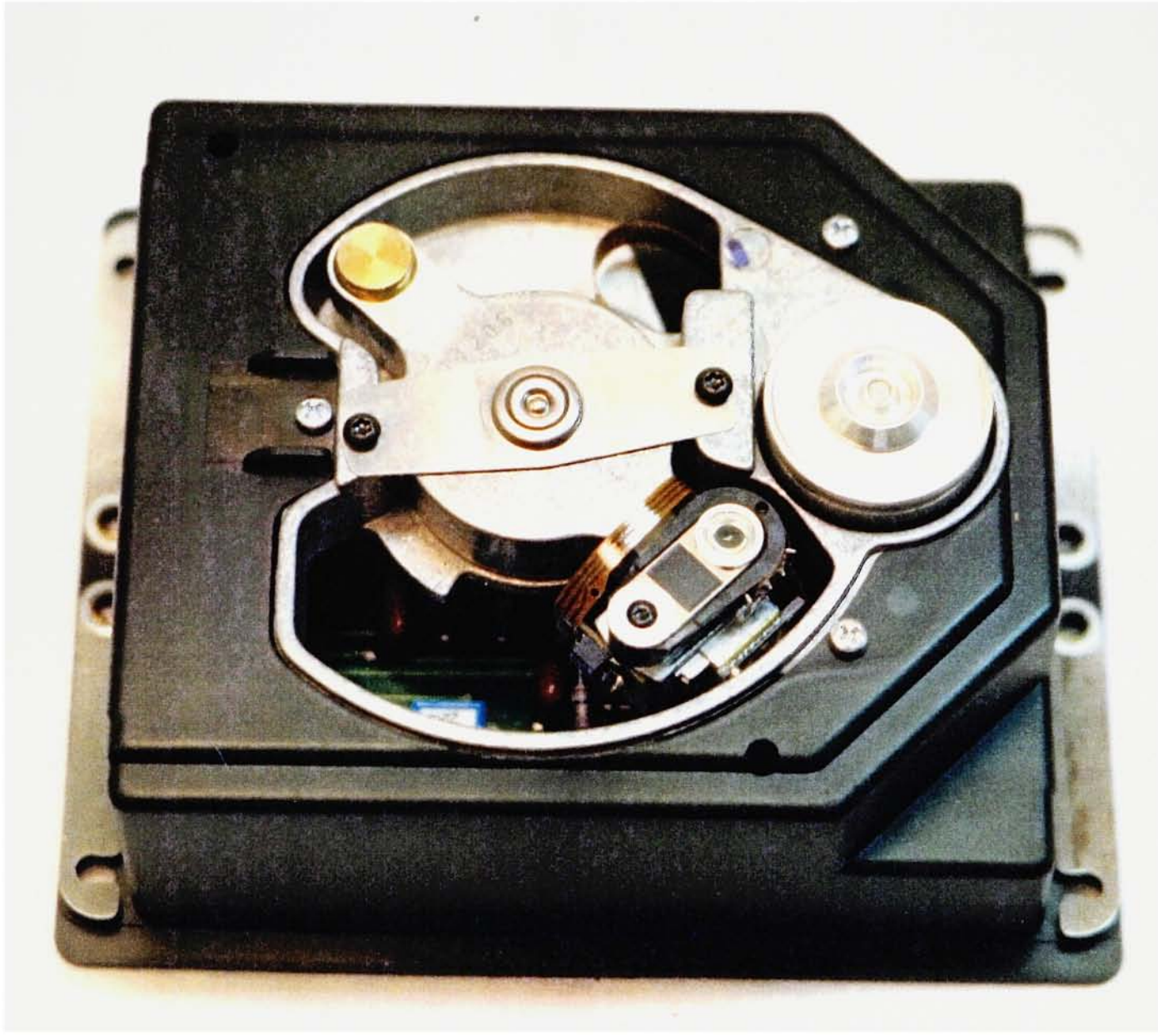
Underside Of Pickup Arm

Plate V. Right Grip Mechanism, Enlarged



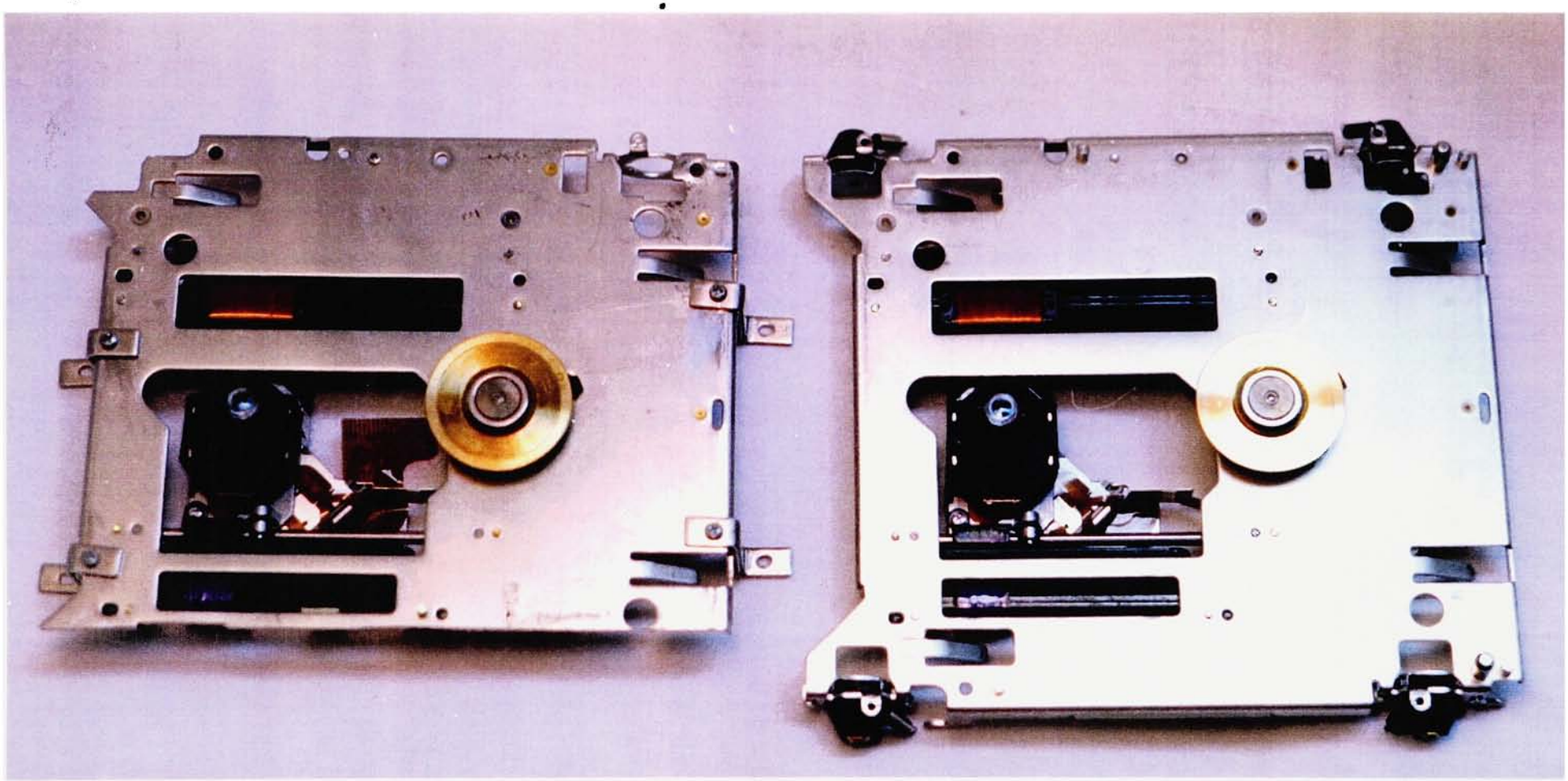
Right Grip Mechanism, Enlarged

Plate VI. Original Audio Compact Disc Player



Original Audio Compact Disc Player

Plate VIII. Modified And Unmodified CD ROM Drives



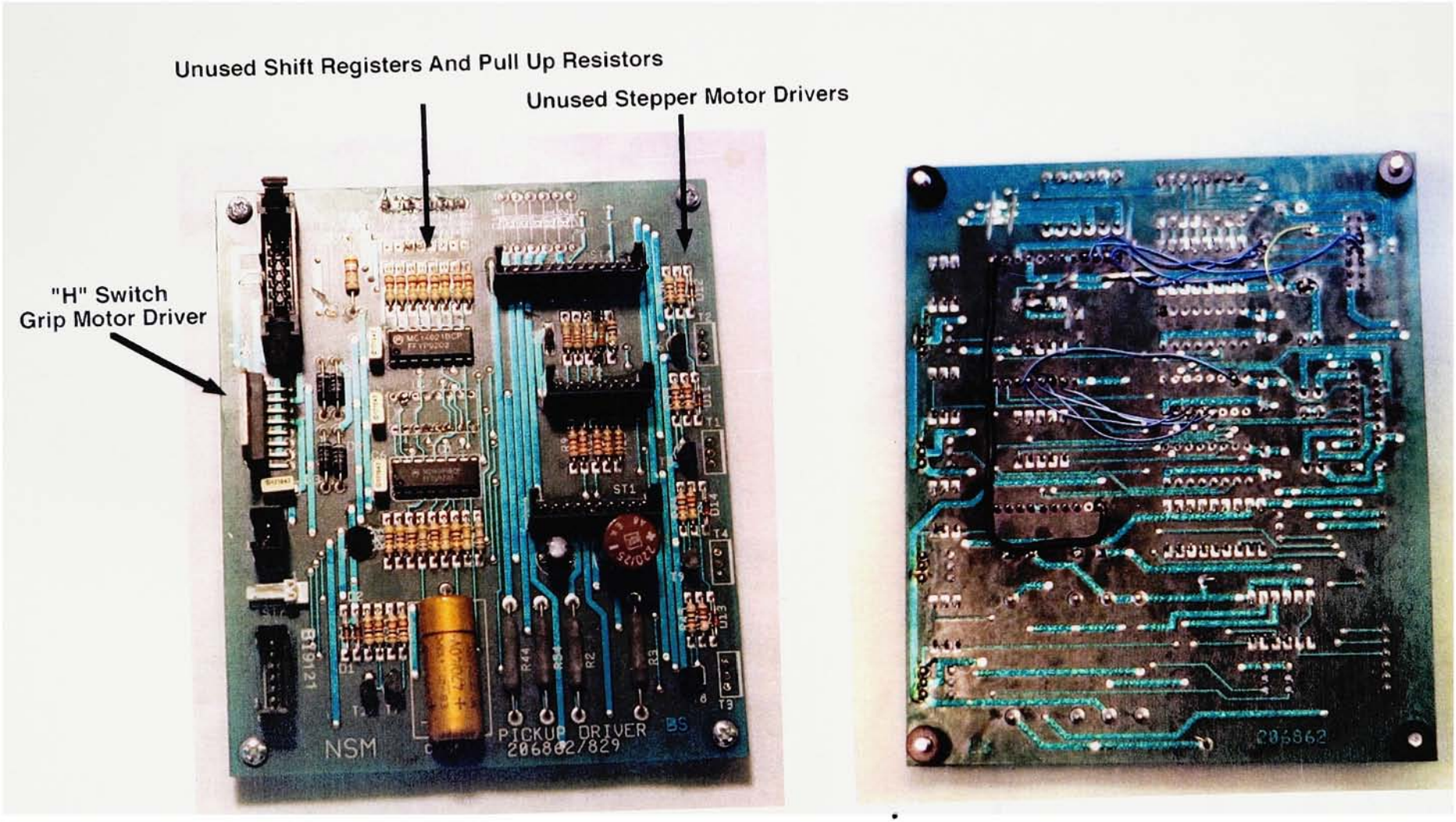
Modified CD ROM Drive

Unmodified CD ROM Drive

Plate IX. Intelligent Driver Device

Intelligent Driver Device

Plate X. Modified Pickup Driver Board



Modified Pickup Driver Board

References

1. Thibeau, Donald. 1992. "CD Imaging Q & A". Eastman Kodak Company.

2. Intel Corporation. 1986. "PL/M 51 User's Guide For DOS Systems".

3. Intel Corporation. 1986. "MCS-51 Assembler User's Guide For DOS Systems".

4. Intel Corporation. 1989. "8-Bit Embedded Controller Handbook".

5. NSM, AG. 1992. "Technical Instruction For NSM-CD 2101 AC".

6. National Semiconductor Corporation. 1990. "Motion Control Handbook".

7. McCracken, Daniel D. 1978. "A Guide To PL/M Programming For Microcomputer Applications".